A Parallel Narrowing Strategy

Michael Hanus

RWTH Aachen

Joint work with

Sergio Antoy

Portland State University

and

Rachid Echahed

IMAG-LSR, Grenoble

Evaluation Principles of Functional Logic Languages

Narrowing:

(ALF, Babel, K-Leaf, LPG, SLOG,...)

Instantiate goal variables and apply a reduction step

$$
\begin{array}{rcl}
\texttt{0+X} & \rightarrow & \texttt{X} \\
\texttt{s(X)+Y} & \rightarrow & \texttt{s(X+Y)}
\end{array}
$$

$$
\begin{array}{lll}
\texttt{Z+s(0) = s(s(0))} & \leadsto_{\{\texttt{Z}\mapsto\texttt{s(X)}\}} & \texttt{s(X+s(0)) = s(s(0))} \\
& \leadsto_{\{\texttt{X}\mapsto\texttt{0}\}} & \texttt{s(s(0)) = s(s(0))}
\end{array}
$$

Computed answer: $\{\texttt{Z} \mapsto \texttt{s(0)}\}$

Advantage: Ability to solve equations

Disadvantage: huge search space

(nondeterministically select a position *and* a rule)

$\Longrightarrow$ **narrowing strategies** [Hanus JLP'94]

Lazy Narrowing

- apply narrowing steps at outermost positions
- evaluate arguments if necessary

$$\texttt{0*X} \quad \to \quad \texttt{0}$$

$$\texttt{0*(s(0)+s(z)) = 0} \quad \leadsto_{\{\}} \quad \texttt{0 = 0}$$

Permits computations with infinite data structures:

$$\begin{array}{rcl}
\texttt{from(N)} & \to & \texttt{[N|from(s(N))]} \\
\texttt{first(0,L)} & \to & \texttt{[]} \\
\texttt{first(s(N),[E|L])} & \to & \texttt{[E|first(N,L)]}
\end{array}$$

$$\begin{array}{ll}
\multicolumn{2}{l}{\texttt{first(X,from(Y)) = [0]}} \\
\leadsto_{\{\}} & \texttt{first(X,[Y|from(s(Y))]) = [0]} \\
\leadsto_{\{\texttt{X}\mapsto\texttt{s(X1)}\}} & \texttt{[Y|first(X1,from(s(Y)))] = [0]} \\
\leadsto_{\{\texttt{Y}\mapsto\texttt{0}\}} & \texttt{first(X1,from(s(Y))) = []} \\
\leadsto_{\{\texttt{X1}\mapsto\texttt{0}\}} & \texttt{[] = []}
\end{array}$$

Computed answer: $\{\texttt{X} \mapsto \texttt{s(0)}, \texttt{Y} \mapsto \texttt{0}\}$

Eager narrowing strategies run into infinite loop!

Needed Narrowing [Antoy/Echahed/Hanus POPL'94]

Idea: **Evaluate only needed subterms**

1. Determine some needed argument position (from the left-hand sides of the rules)
2. If actual argument is a
   - constructor: proceed with matching
   - function call: evaluate it
   - logical variable: (nondeterministically) instantiate it

$$\begin{array}{rcl} \mathtt{0} \leq \mathtt{X} & \rightarrow & \mathtt{true} \\ \mathtt{s(X)} \leq \mathtt{0} & \rightarrow & \mathtt{false} \\ \mathtt{s(X)} \leq \mathtt{s(Y)} & \rightarrow & \mathtt{X} \leq \mathtt{Y} \end{array} \qquad \begin{array}{rcl} \mathtt{0+X} & \rightarrow & \mathtt{X} \\ \mathtt{s(X)+Y} & \rightarrow & \mathtt{s(X+Y)} \end{array}$$

$$\begin{array}{rlll} \mathtt{X} \leq \mathtt{Y+Z} & \mapsto_{\{\mathtt{X}\mapsto\mathtt{0}\}} & \mathtt{0} \leq \mathtt{Y+Z} & \rightarrow \quad \mathtt{true} \\ \\ \mathtt{X} \leq \mathtt{Y+Z} & \mapsto_{\{\mathtt{X}\mapsto\mathtt{s(X1)}\}} & \mathtt{s(X1)} \leq \mathtt{Y+Z} & \\ & \mapsto_{\{\mathtt{Y}\mapsto\mathtt{0}\}} & \mathtt{s(X1)} \leq \mathtt{0+Z} & \rightarrow \\ \mathtt{s(X1)} \leq \mathtt{Z} & \mapsto_{\{\mathtt{Z}\mapsto\mathtt{0}\}} & \mathtt{s(X1)} \leq \mathtt{0} & \rightarrow \quad \mathtt{false} \end{array}$$

Properties of Needed Narrowing

- **Soundness** and **Completeness** w.r.t. strict equality

- **Optimality:**

  1. **No unnecessary steps:**
     Each narrowing step is needed, i.e., it cannot be avoided if a solution should be computed.

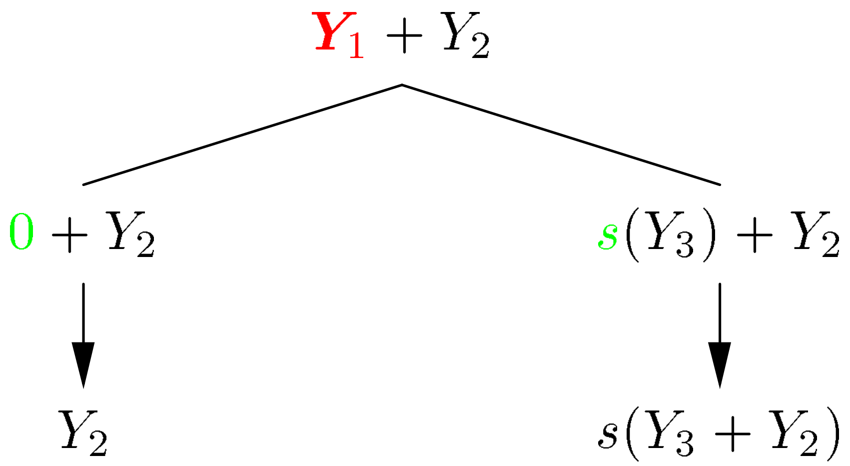
  2. **Shortest derivations:**
     If common subterms are shared, needed narrowing derivations have minimal length.

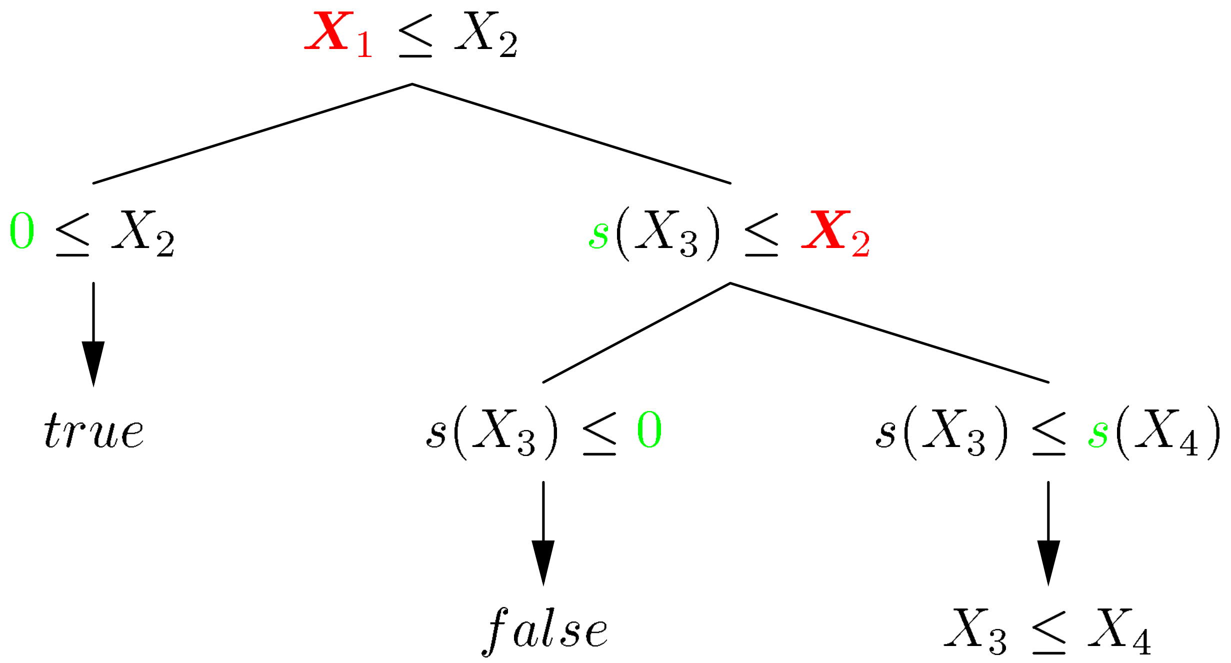
  3. **Independence of solutions:**
     Two solutions σ and σ' computed by two distinct derivations are independent (i.e., $\sigma(x)$ and $\sigma'(x)$ are not unifiable for some variable x).

- **Restriction:** Needed narrowing defined for **Inductively Sequential Rewrite Systems:**
  The rules of each operation can be organized in a *definitional tree*
  (in particular, non-overlapping left-hand sides)

Definitional Trees [Antoy 92]

- data structure to organize the rules of an operation
- each node has a distinct *pattern*
- *branch* nodes (case distinction), *rule* nodes





Overlapping Rules: Problems

$$\begin{array}{rcl} \texttt{X} \vee \texttt{true} & \to & \texttt{true} \\ \texttt{true} \vee \texttt{X} & \to & \texttt{true} \\ \texttt{false} \vee \texttt{false} & \to & \texttt{false} \end{array}$$

Evaluate $t_1 \vee t_2$: narrow t_1 or narrow t_2?

Solution in current languages:
Backtracking: narrow t_1, if not successful: narrow t_2

Disadvantage:

- unnecessary evaluations
- incompleteness because of infinite narrowing in t_1
- non-determinism even for ground terms!
  (Backtracking on ground terms: Functional programming?)

Our solution:
Apply evaluation steps to t_1 and t_2 in parallel

Parallel Narrowing Steps

1. Compute all narrowing substitutions
   (for all overlapping cases)

2. Minimize substitution set

3. Perform parallel rewriting for each substitution

$$\begin{array}{rcl} \mathtt{X} \vee \mathtt{true} & \to & \mathtt{true} \\ \mathtt{true} \vee \mathtt{X} & \to & \mathtt{true} \\ \mathtt{false} \vee \mathtt{false} & \to & \mathtt{false} \end{array} \qquad \mathtt{f(a)} \to \mathtt{true}$$

$\mathtt{f(X)} \vee \mathtt{f(X)}$:

1. Computed substitutions: $\{\mathtt{X} \mapsto \mathtt{a}\}$ and $\{\mathtt{X} \mapsto \mathtt{a}\}$

2. Minimized substitution set: $\{\ \{\mathtt{X} \mapsto \mathtt{a}\}\ \}$

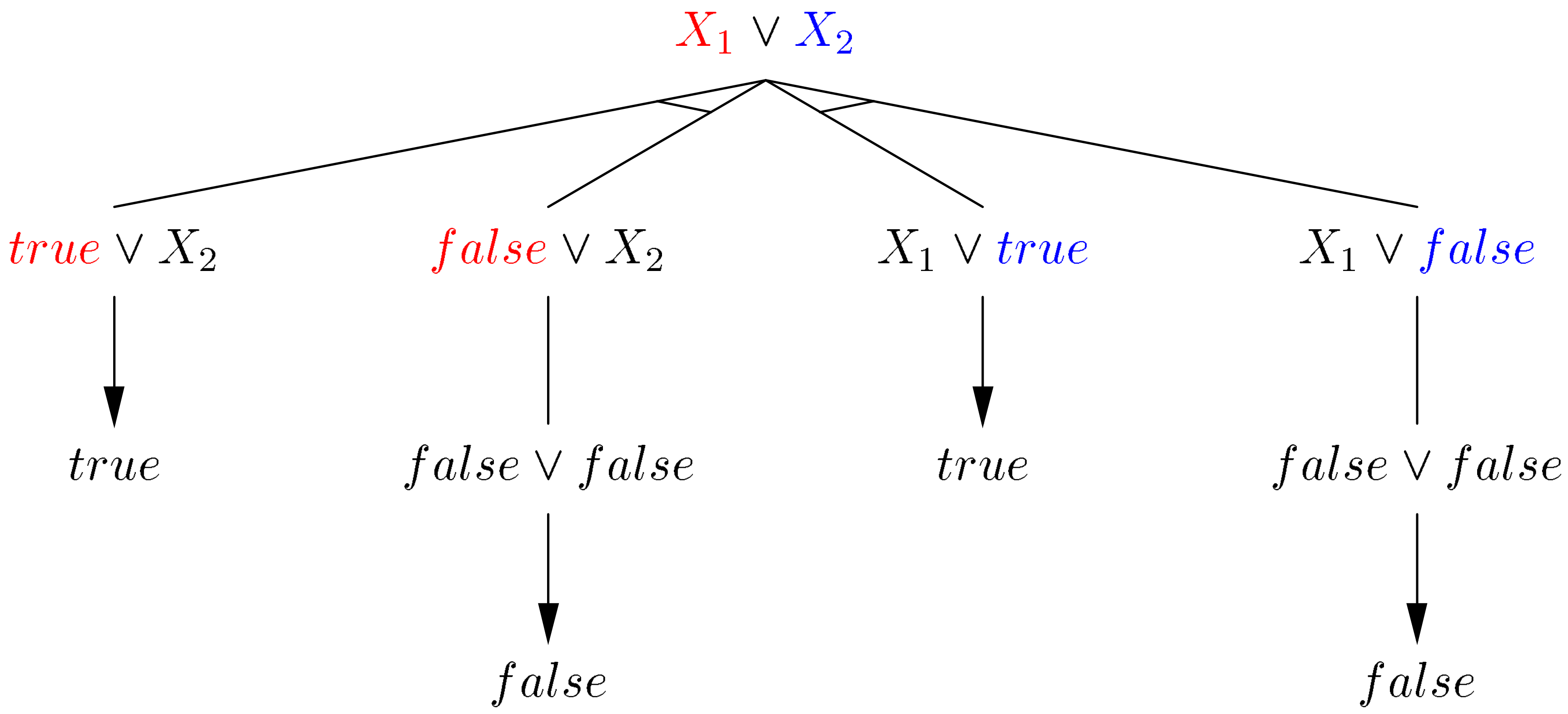
3. Apply substitution + parallel rewriting:

   $\mathtt{f(X)} \vee \mathtt{f(X)} \ \leadsto_{\{\mathtt{X} \mapsto \mathtt{a}\}} \ \mathtt{true} \vee \mathtt{true} \ \leadsto_{\{\}} \ \mathtt{true}$

Standard lazy narrowing computes four distinct derivations!

Parallel Definitional Trees

May contain branch nodes with more than one branching argument:



Parallel Narrowing: Minimize Substitutions

- Prefer substitutions belonging to outer subterms
- Take minimal substitutions among the non-identity computed substitutions

$$\begin{array}{rclcrcl} \texttt{f(X,s(Y))} & \to & \texttt{s(0)} & \quad & \texttt{h(s(s(X)))} & \to & \texttt{s(0)} \\ \texttt{f(0,Y)} & \to & \texttt{s(0)} & & \texttt{g(X)} & \to & \texttt{s(g(X))} \end{array}$$

`f(g(X),f(h(Y),f(0,h(s(Y))))):`

1. Computed position/substitution pairs:
   $\{(1, \{\}), (2\cdot 1, \{\texttt{Y} \mapsto \texttt{s(s(Y1))}\}), (2\cdot 2, \{\}), (2\cdot 2\cdot 2, \{\texttt{Y} \mapsto \texttt{s(Y2)}\})\}$
2. Minimized substitution set: $\{\ \{\}\ \}$

Consider non-identity substitutions in the presence of non-terminating rules:

$$\begin{array}{rclcrcl} \texttt{f(X,0)} & \to & \texttt{s(0)} & \quad & \texttt{h(0)} & \to & \texttt{0} \\ \texttt{f(0,X)} & \to & \texttt{s(0)} & & \texttt{g(X)} & \to & \texttt{g(s(X))} \end{array}$$

$$\texttt{f(g(X),h(Y))} \ \overset{\rightarrow}{\approx}_{\{\texttt{Y}\mapsto\texttt{0}\}} \ \texttt{f(g(s(X)),0)} \ \overset{\rightarrow}{\approx}_{\{\}} \ \texttt{0}$$

Properties of Parallel Narrowing

- **Soundness** and **Completeness**
  w.r.t. strict equality

- **Deterministic evaluation** of ground terms

- Computes always a normal form if it exists

- **Optimality:**
  - Equivalent to *needed narrowing* if the occurring functions are inductively sequential
    ($\approx$ optimal on non-overlapping parts)

  - Equivalent to *rewriting with necessary sets* [Sekar/Ramakrishnan 93] on ground terms
    ($\approx$ optimal if rhs are not considered)

Related Work

Improving narrowing strategies for overlapping rules:

- **Dynamic cut** [Loogen/Winkler PLILP'91]:
  Ignore alternative rules if no goal variable bound
  $\Rightarrow$ same effect by parallel narrowing

- **Lazy narrowing with simplification** [Hanus 94]:
  Perform deterministic lazy rewrite steps before narrowing steps
  $\Rightarrow$ subsumed by parallel narrowing, but only for terminating systems!

$$\begin{array}{lll} \texttt{f(0,X,Y)} & \to & \texttt{0} \\ \texttt{f(X,0,Y)} & \to & \texttt{0} \\ \texttt{f(X,Y,0)} & \to & \texttt{0} \end{array} \qquad \begin{array}{lll} \texttt{g(X)} & \to & \texttt{g(s(X))} \\ \texttt{h(0)} & \to & \texttt{0} \end{array}$$

$$\texttt{f(g(X),h(Y),h(0))} \;\to\; \texttt{f(g(X),h(Y),0)} \;\to\; \texttt{0}$$

Substitutions computed by parallel narrowing:
$\{\ \{\}, \{\texttt{Y} \mapsto \texttt{0}\}\ \}$ $\leadsto$ non-deterministic step

$\Longrightarrow$ Combine parallel narrowing with simplification

Conclusions

- Narrowing strategy for constructor-based rewrite systems with overlapping rules

- Termination not required
  ($\sim$ functional programming)

- Idea: parallel evaluation of necessary sets of redexes

- Minimization of substitutions
  $\leadsto$ reducing non-deterministic steps

- Conservative extension of two optimal strategies:

  – needed narrowing

  – rewriting with necessary sets

- First narrowing strategy for this class with deterministic evaluation of ground terms

- Efficient implementation by local computations based on unification

$\leadsto$ Ideal strategy for functional logic languages